

ASSISTANT COMMISSIONER FOR PATENTS
BOX PATENT APPLICATION
WASHINGTON, D.C. 20231

04-14-00
CASE DOCKET NO. YOR9-2000-0185-US1
DATE: April 13, 2000

Sir:

Transmitted herewith for filing under rule 1.53(f) is the Patent Application of:

Inventors: Douglas Lee Schales, Srinivasan Seshan and Miriam Zohar.

For: NETWORK DATA PACKET CLASSIFICATION AND DEMULTIPLEXING

Enclosed are:

7 Sheets of formal Drawings.

An assignment of the invention to International Business Machines Corporation, Armonk, New York 10504.

A certified copy of a _____ application.

Declaration and Power of Attorney is attached to the application.

Associate Power of Attorney.

Information Disclosure Statement with form PTO-1449 with references attached.

The filing fee has been calculated as shown below:

	(Col. 1)	(Col. 2)
FOR:	NO. FILED	NO. EXTRA
BASIC FEE		
TOTAL CLAIMS	39 - 20 =	19
INDEP CLAIMS	6 - 3 =	3
____ MULTIPLE DEPENDENT CLAIM PRESENTED		

OTHER THAN A
SMALL ENTITY

RATE	FEE
	\$ 690.00
X \$ 18 =	\$ 342.00
X \$ 78 =	\$ 234.00
+ \$ 260 =	\$ 0.00
TOTAL	\$ 1,266.00

If the difference in Col. 1 is less than zero, enter "0" in Col. 2.

☒ Please charge my Deposit Account No. 09-0468 in the amount of \$1,266.00.

☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 09-0468. A duplicate copy of this sheet is enclosed.

☒ Any additional filing fees required under 37 CFR 1.16.

☒ Any patent application processing fees under 35 CFR 1.17.

Respectfully submitted,

By

Louis Herzberg
Louis P. Herzberg
Registration No.: 41,500
Tel. (914) 945-2885

IBM CORPORATION
INTELLECTUAL PROPERTY LAW DEPT.
P.O. BOX 218
YORKTOWN HEIGHTS, NY 10598

Express Mail Label No.:
EL559660789US
Date of Deposit: April 13, 2000



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Schales et al.

Docket No.: YOR9-2000-0185-US1

Serial No.:

Group No.:

Filed: Herewith

Examiner:

For: NETWORK DATA PACKET CLASSIFICATION AND DEMULTIPLEXING

Assistant Commissioner for Patents

Washington, D.C. 20231

EXPRESS MAIL CERTIFICATE

Express Mail Label Number EL559660789US

Date of Deposit April 13, 2000

I hereby certify that the attached paper or fee

Patent Application Transmittal Letter (original and one copy)

Patent Application Under Rule 1.53(f) (Missing Parts)

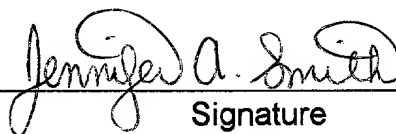
Formal Drawings (7 Sheets)

Return Postcard

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Jennifer A. Smith

(Name)



Signature

Note: Each paper must have its own certificate and the "Express Mail" label number as a part thereof or attached thereto. When, as here, the certification is presented on a separate sheet, that sheet must (1) be signed and (2) fully identify and be securely attached to the paper or fee it accompanies. Identification should include the serial number and filing date of the application as well as the type of paper being filed, e.g. complete application, specification and drawings, responses to rejection or refusal, notice of appeal, etc. If the serial number of the application is not known, the identification should include at least the name of the inventor(s) and the title of the invention.

Note: The label number need not be placed on each page. It should, however, be placed on the first page of each separate document, such as, a new application, amendment, assignment, and transmittal letter for a fee, along with the certificate of mailing by "Express Mail". Although the label number may be on checks, such a practice is not required. In order not to deface formal drawings it is suggested that the label number be placed on the back of each formal drawing or the drawings be accompanied by a set of informal drawings on which the label number is placed.

1 NETWORK DATA PACKET CLASSIFICATION AND DEMULTIPLEXING

2 FIELD OF THE INVENTION

3 The present invention is directed to the field of packet
4 communication. It is more particularly directed to
5 classification and demultiplexing of network communication
6 packets processed in a network protocol stack.

7 BACKGROUND OF THE INVENTION

8 Communication over a network often requires the information that
9 is to be transported from one computer to another be divided into
10 network communication packets. These network communication
11 packets, simply referred to as "packets", are transported across
12 the physical communication network.

13 The information originating from an application program becomes
14 packetized into network communication packets by passing through
15 various software components before arriving at the network
16 interface card for transmission on the physical communications
17 network. These software components are typically layered to form
18 what is known as the network protocol stack. Each layer is
19 responsible for a different facet of communication. For example,
20 the TCP/IP protocol stack is normally split into four layers:
21 link, network, transport and application. Figure 1 shows the
22 relationship between the protocol layers and the TCP/IP protocol
23 stack. The link layer 101 is responsible for placing data on the

1 physical network. The network layer 102 is responsible for
2 routing. The transport layer 103 is responsible for the
3 communication between two hosts. The application layer 104 is
4 responsible for processing the application specific data.

5 For example, Figure 2 illustrates the stages of an HTTP request
6 being encapsulated before being sent to a web server. As the
7 request descends the protocol stack, each layer 201-204
8 encapsulates the packet adding its own header. When the HTTP
9 packet arrives at the destination address, each protocol layer
10 uses information within its header to classify the incoming
11 packet amongst all the protocols in the layer above it. This
12 process is commonly referred to as demultiplexing.

13 At each layer in the network protocol stack, the packet is
14 demultiplexed or "classified" based on information about the
15 packet that is contained in the headers or from information
16 inside the data portion of the packet itself. The packet is
17 processed differently based on its classification.

18 For example, Figure 3 illustrates how this classification is done
19 for an incoming HTTP request 301. The Ethernet driver 302, in
20 the link layer 300, classifies the packet based on frame type in
21 the Ethernet header and passes it to IPv4 312 in the network
22 layer 310. IPv4 312 classifies the packet based on the IP header
23 protocol value in the IP header and passes it to TCP 323 in the
24 transport layer 320. TCP classifies the packet based on the
25 destination port number in the TCP header and passes it to the
26 HTTP server 332 in the application layer 330.

1 Traditional packet classification systems, as found in BPF, DPF,
2 Pathfinder, Router Plugins, operating systems and many firewalls,
3 are limited to a set of fixed pattern matching rules. This
4 allows a user to intercept/process any packet that matches the
5 desired set of values in the appropriate byte ranges (usually a
6 combination of the IP and the protocol header fields, such as
7 source/destination address, protocol or source/destination
8 ports). These packets are then passed to a software module that
9 processes the packets and can modify, forward, drop or delay
10 them. Stateful packet filtering systems generally have the
11 ability to generate and add rules dynamically based on
12 application traffic. However, such systems do not provide simple
13 methods to extend packet processing to understand new application
14 protocols.

15 These traditional systems may work well for applications that use
16 a single connection to a well known destination address and port.
17 However, many modern applications initially use a well known
18 service port for the control session and then use additional
19 connections on ephemeral port numbers for each data stream.
20 Examples of such applications are FTP, Real Audio and H.323. To
21 support these applications efficiently, the traditional systems
22 must allow packet matching filter rules to be updated dynamically
23 and quickly. In addition, some modern protocols have abandoned
24 using fixed format headers and fixed sized fields. For example,
25 HTTP makes its header human readable by encoding them as
26 strings.

27 SUMMARY OF THE INVENTION

1 It is thus an aspect of the present invention to provide greater
2 flexibility in classifying and demultiplexing packets in the
3 network protocol stack. As a result, it provides a method for
4 application level classification. This is due to classifying
5 techniques and a modular structure described subsequently.

6 Another aspect of the present invention provides easier
7 extendibility for packet processing in the network protocol stack
8 by defining a standard method for adding new functionality or
9 support for new protocols and applications.

10 Another aspect of the present invention provides methods and
11 apparatus to obtain external information, from an application
12 scheduled outside of the forwarding or interrupt context of the
13 kernel, in order to augment packet classification and/or
14 disposition.

15 An example embodiment of the present invention is a method for
16 classifying a data packet. The method includes the steps of:
17 receiving the packet at a root node of a classification tree;
18 passing the packet to a first child node of a first tree level of
19 the classification tree indicating a satisfaction of a
20 node-criteria of the first child node; the first child node
21 forming the data packet into a matched packet; and repeating the
22 step of passing and forming for a next tree level until no first
23 child node of the next level at a succeeding next level indicates
24 satisfaction of the node-criteria of the first child node of the
25 next level.

1 In some embodiments the step of indicating includes the step of
2 executing a set of code which returns a status indication of the
3 type; and/or the step of indicating satisfaction of a criteria
4 includes the steps of executing a set of code which identifies
5 the desired packet and returning a status indication; and/or the
6 step of forming the data packet into a matched packet includes
7 the step of indicating satisfaction; and/or the step of repeating
8 the step of passing and the step forming includes the steps of
9 indicating and returning a status indication of NO_Match.

10 In some embodiments of the method, the method further includes:
11 the step of adding at least one new child node; and/or one new
12 child node is a Real Audio node; and/or the method is extendible
13 such that one or more nodes are dynamically added at any level;
14 parsing the matched packet and generating relevant information;
15 transforming the matched packet into a transformed packet; and/or
16 associating the packet at a last first child node indicating
17 satisfaction; executing a set of code in accordance with the last
18 first child node; and/or the step of forming includes the first
19 child node specifying a set of code to be run subsequently;
20 and/or the step specifying specifies the set of code to be run
21 following classification.

22 Another example embodiment of the present invention is a method
23 which uses an external process for classifying a packet. This
24 method includes the steps of suspending a classification process
25 in progress for the packet, and obtaining external information
26 employed in the classifying. This is performed by an
27 application scheduled outside of the forwarding or interrupt
28 context of the kernel.

1 In some embodiments of the method, the step of suspending
2 includes the steps of queuing any data, including information
3 about the packet or its present classification; and/or
4 transferring said data to an application that is scheduled
5 outside of the forwarding or interrupt context of the kernel.

6 In some embodiments of the method, the step of obtaining external
7 information includes augmenting a node-criteria of a node in a
8 classification tree with additional information; and/or the
9 external information includes authentication of an originator of
10 the packet; the classification process is an extendible
11 classifier process (In one application, a process is extendible
12 by adding a new child node); and/or the step of specifying
13 includes enforcement of a site policy. A site policy is
14 composed of a number of different aspects including security.
15 The security aspect of a site policy may be based on packet
16 classification and authentication information.

17 Another aspect of the present invention is a method for
18 determining disposition of an original packet received at a child
19 node. The method includes the step of passing the original
20 packet and a first disposition of the original packet to an
21 external process, and the external process augmenting the
22 original packet and/or augmenting the first disposition by
23 employing a process specific means and returning an augmented
24 packet and an augmented disposition to the child node. Some
25 embodiments of the method include suspending a disposition
26 process in progress for the original packet; and/or the augmented

1 disposition includes identification of and/or authentication of
2 an originator of said packet.

3 **BRIEF DESCRIPTION OF THE DRAWINGS**

4 These and other aspects, features, and advantages of the present
5 invention will become apparent upon further consideration of the
6 following detailed description of the invention when read in
7 conjunction with the drawing figures, in which:

8 Fig. 1 shows the relationship between the protocol layers and the
9 TCP/IP protocol stack;

10 Fig. 2 illustrates the stages of an HTTP request being
11 encapsulated before being sent to a web server;

12 Fig. 3 illustrates how classifying is done for an incoming HTTP
13 request;

14 Fig. 4 shows an example of how to organize modules in the
15 classification tree in accordance with the present invention;

16 Fig. 5 shows an example of a packet classification and
17 demultiplexing process in classifying a packet in accordance with
18 the present invention;

19 Fig. 6 shows an example of steps to determine the packet
20 disposition in accordance with the present invention;

1 Fig. 7 shows an example of pm_t return codes in accordance with
2 the present invention;

3 Fig. 8 shows an example of application dependent nodes in
4 accordance with the present invention;

5 Fig. 9 shows an example of pp_t return codes in accordance with
6 the present invention;

7 Fig. 10 shows an example of paction_t return codes in accordance
8 with the present invention; and

9 Fig. 11 shows an example of an apparatus in accordance with the
10 present invention.

11 DETAILED DESCRIPTION OF THE INVENTION

12 Networking protocols are normally divided into layers which are
13 responsible for different facets of communication as Figure 1
14 depicts for the network layers of the TCP/IP protocol. The
15 associated call graph created by the standard UNIX protocol stack
16 is arranged like a tree as described for Figure 3. Each level of
17 the tree corresponds to a different layer in the networking
18 protocol stack. The present invention mimics the call graph of
19 the UNIX protocol stack and organizes the different modules
20 competing for packets at the IP layer in a tree structure herein
21 referred to as a classification tree.

1 An example of a classification tree 400 is shown in Figure 4.
2 Figure 4 shows each node in the classification tree as a separate
3 module. In an embodiment of the present invention each node is
4 composed of 4 packet traversal functions (matcher, preprocessor,
5 action, and post processing) and 3 node management functions
6 (callback, heartbeat and management). Only the packet matching
7 function, which identifies the packets to process, and the packet
8 action function, which determines the packet disposition, are
9 required. The packet matching function is herein referred to as
10 the node-criteria of the node. The remaining traversal and
11 management function pointers can default to NULL. These
12 functions associated with each node are stored in a PacketFilter
13 structure.

14 Since each of the nodes is a separate dynamically loadable
15 module, the classification tree organization is flexible. In an
16 embodiment of the present invention, the modules are loaded into
17 memory during the initialization process. Based upon
18 configuration information the modules are then arranged to form a
19 classification tree. The ordering of the modules is important
20 since the packet traversal is governed by this ordering. As the
21 classification tree is created, each node is initialized by
22 executing a set of code. In the embodiment, this set of code is
23 a function referred to as the management function(*mm*). The input
24 parameter to the *mm* function is generally a single pointer to a
25 buffer containing the node specific configuration data.

26 Figure 4 shows an example of how to organize modules in the
27 classification tree. The IPv4 503, IPv6 504, UDP 506, HTTP 507
28 and TCP 508 modules each wish to observe or modify packets that

1 use the protocol for which they are named. However, in this
2 example, there are multiple ways one could imagine wanting to
3 process HTTP requests. These ways include: providing a
4 transparent HTTP proxy function, using a specialized TCP for HTTP
5 like Transaction TCP(T/TCP), doing content filtering based on
6 site policy, or limiting packet traffic based on a service
7 contract. Depending on the intended purpose of the
8 classification tree, differing modules are loaded into memory. A
9 site policy is composed of a number of different aspects
10 including security. The security aspect of a site policy may be
11 based on packet classification and authentication information.
12 Once initialization completes, the classification tree may be
13 modified by adding, deleting or moving a node. This ability of
14 modifying the classification tree makes the packet classification
15 process extendible.

16 The present invention includes methods for implementing a packet
17 classification process and/or an augmented packet disposition
18 process. The packet to be classified and/or augmented is herein
19 referred to as the original packet. The resulting packet is
20 referred to as the augmented packet. The disposition of the
21 original packet is herein referred to as the first disposition,
22 and the disposition resulting from the augmented disposition
23 process is herein referred to as the augmented disposition.
24 Anything outside of the forwarding or interrupt context of the
25 kernel is herein said to be external.

26 An example embodiment has 7 steps to classify a packet and
27 determine the augmented packet disposition. These steps are in
28 the interrupt context except where noted. Steps 1-4 describe the

1 packet classification process shown in Figure 5. Steps 5-7
2 describe augmenting the packet disposition process. A flow
3 diagram for these seven steps is shown in Figure 6. Refer to
4 figures 5 & 6 for the following description.

5 Step 1: After receiving a packet from the physical network, the
6 Link Layer passes the packet to the root node 502.

7
8 In this step, a network driver receives a packet from the
9 physical network, which it classifies based on frame type in
10 the MAC header and passes it to the root node of the
11 classification tree.

12
13 Step 2: The packet is passed to a first child node of the first
14 level 521 of the classification tree, indicating a satisfaction
15 of a node-criteria of the child node.

16 The root node asks each child node from left to right whether
17 the packet matches its node-criteria, until a child node's
18 node-criteria is satisfied. The root node then passes the
19 packet to that first child node which satisfies the
20 node-criteria and forms the data packet into a matched packet.
21 In Figure 5, the root node 502 first passes the packet to the
22 IPv4 node 503. A child node's node-criteria includes a set
23 of code used to identify the packets desired. This set of
24 code is implemented as a function referred to as the packet
25 matching function (*pm*) 603.

26 The input parameters to the *pm* function are: the PBUF, an
27 operating system independent data structure containing the

1 packet, the options memory area, and a pointer to the packet
2 filter node. The result of the packet matching function,
3 indicating satisfaction or lack thereof, of the child node's
4 node-criteria, is of type *pm_t*. Figure 7 enumerates a sample
5 group of type *pm_t* return code values 700. The packet
6 matching function results indicating satisfaction of a child
7 node's node-criteria include: Match_OK, Match_This,
8 Match_Discard, and Match_Forward. The result indicating lack
9 of satisfaction is NO_Match.

10 The packet matching function may be as simplistic as matching
11 a static fixed offset, such as the IPv4 node, or as complex as
12 identifying the packets for applications which negotiate
13 additional connections, such as FTP, Real-Audio and H.323.
14 Unfortunately, since each of these applications has its own
15 method for negotiating additional connections, application
16 dependent nodes are required. This is as illustrated in
17 Figure 8 for H.323 831, Real-Audio 832, and FTP 833. For each
18 additional connection, a dynamic filter rule is created. These
19 dynamic filter rules and other state information for the
20 negotiated connections are stored locally in the application
21 specific node. One implementation uses a hashtable structure
22 for storing this data. Based on the well known services port
23 and the application specific data, the packet matching
24 function identifies the packets desired enabling application
25 level classification.

26 Step 3: Repeat the process of 'passing the packet' starting with
27 a first child node of a next tree level of the classification
28 tree which satisfies a node-criteria of that first child node, as

1 described in step 2, and form the packet into a matched packet,
2 until no child of a next tree level of the classification tree
3 succeeds in satisfying a node-criteria (No_Match).

4 A determination is made if there is a next child 604. If
5 there is, flow continues with 601. If not, flow continues
6 with 621. Thus, when the packet matching function of all of
7 the children nodes of the next tree layer result in a lack of
8 satisfaction, (No_Match), the packet is said to have fully
9 traversed the classification tree. The traversal path is
10 defined as the set of nodes from the root to the last first
11 child node satisfying a node-criteria of the child node. Thus
12 packet classification has completed and flow continues with
13 621.

14 Step 4: For each first child node, satisfying a node-criteria of
15 the child node form the data packet into a matched packet. This
16 may be performed as in steps 4A, 4B and/or 4C.

17
18 Step 4A: The current node is added to the node
19 traversal path 605.

20
21 Step 4B: The node may execute a set of code, if such a
22 code exists, which may parse and transform the
23 packet 607. If the set of code exists, the packet
24 is parsed and transformed 617. If not, flow
25 continues with 609.

26 Once a node's node-criteria is satisfied, the packet's
27 traversal of the classification tree is limited to the

1 node's descendants. The remainder of the classification
2 tree is not traversed. But before traversing a node's
3 subtree, the node may execute a set of code. In the
4 present embodiment, this set of code is referred to as
5 the packet preprocessor function(pp). The input
6 parameters are the same as the packet matching function.
7 This includes: the PBUF, an operating system independent
8 data structure containing the packet, the options memory
9 area, and a pointer to the packet filter node. The
10 return code of the pp function is of type pp_t. Examples
11 of type pp_t 900 are enumerated in Figure 9. The packet
12 preprocessor function may perform actions such as parsing
13 a packet and transforming a packet. Parsing a packet
14 generates information that may need to be made available
15 to the node's descendants and ancestors. Transforming a
16 packet takes place for example, when the IPSEC node's
17 preprocessor transforms an encrypted packet into a
18 decrypted packet. IPsec tunnel information and other
19 information is created that can be used by other nodes in
20 the classification tree.

21 The present invention thus provides a generic mechanism
22 for retaining or passing state information between nodes
23 by a mechanism we referred to herein as options passing.
24 In an example of option passing, an options memory
25 segment is attached to each packet during its tree
26 traversal. Each node may store and retrieve state using
27 the APIs: fw_add_option, fw_next_option. Since a node
28 may not understand all options that are passed to it, the

1 node will process the options it understands and ignore
2 those which it does not understand.

3 Step 4C: The node may also suspend the classification
4 process in order to obtain additional external information
5 so as to augment packet classification and demultiplexing
6 615.

7 Suspending the classification process involves queueing
8 any data, including information about the packet or its
9 present classification, and transferring the data to an
10 application that is scheduled outside of the forwarding
11 or interrupt context of the kernel.

12 One embodiment augments packet classification by
13 suspending the packet classification process until the
14 application, scheduled outside of the forwarding or
15 interrupt context of the kernel, completes. The
16 resulting external information is used to augment the
17 packet classification.

18 Examples of applications which may augment packet
19 classification include packet identification and
20 authentication agents. An identification/authentication
21 agent, may use s/ident for out of band identification and
22 authentication. Authentication may use s/ident for out
23 of band authentication in order to correlate the packet
24 with a userid. Another example of authentication is to
25 correlate a VPN tunnel id with a userid.

1 This external information, such as packet
2 identification and/or authentication, permits packets to
3 be handled differently. For example, assume that a site
4 connected to the Internet is severely bandwidth limited.
5 As a result only a limited number of employees at any
6 given moment can run applications with high bandwidth
7 demands, such as streaming media. Based on the external
8 information a site policy can be implemented which gives
9 preferential treatment to a set of employees.

10 Step 5: After packet classification completes, a set of code
11 associated with the last child node which satisfied the
12 node-criteria, is executed.

13
14 In an embodiment, this set of code is referred to as the
15 packet action function(*pa*). The packet action input
16 parameters are: the PBUF, a pointer to the node, a pointer to
17 the node traversal path, and the options memory area. An
18 example of return codes of the *pa* function, of type *paction_t*
19 are enumerated in Figure 10 1000. The return code obtained
20 determines the packet disposition.

21 Normally the packet action function 621 monitors the packet
22 data in order to obtain application specific state information
23 used by the other node functions. For example, a packet
24 action function with application specific knowledge could
25 monitor the packet data for new negotiated data connections.
26 These new dynamic connections are stored locally in the
27 application specific node. The packet matching function uses

1 the dynamic data as part of the node-criteria for application
2 level packet classification.

3 Other examples of packet action function usage include:
4 modifying packets, which may be used to implement NAT;
5 queueing packets, which may be used to shape traffic; dropping
6 packets, which may be used for rate limiting; and redirecting
7 packets, which may be used for load balancing.

8
9 The packet action function may also suspend kernel packet
10 processing and transfer any data (including information about
11 the packet or its classification) to an application that is
12 scheduled outside of the forwarding or interrupt context of
13 the kernel, and/or to obtain external information in order to
14 augment the packet disposition (i.e. discard, forward, process
15 locally or redirect) decision 631. Suspending the packet
16 disposition decision process involves queueing any data,
17 including information about the packet or its classification,
18 and transferring the data to an application employing a
19 process specific means that is scheduled outside of the
20 forwarding or interrupt context of the kernel.

21 An example method of augmenting the packet disposition
22 decision is to suspend any in progress packet disposition
23 decision process until the application scheduled outside of
24 the forwarding or interrupt context of the kernel completes.
25 The resulting external information is used to augment the
26 packet disposition decision. Examples of applications which
27 may augment the packet disposition decision are policy
28 enforcement and content filtering agents based on any

1 combination of the packet classification, identification and
2 authentication. Examples of process specific means include
3 s/identd and external LDAP servers.

4 Once the application completes, it passes the original data,
5 the external information and the results to the kernel, which
6 issues a call to the node's callback function. The callback
7 function(cb) reinserts the packet at the node which suspended
8 the processing. Based on the application's results, dynamic
9 rules may be created 621.

10 For example, an especially advantageous usage is with VPN
11 tunnels. Differing policies based on the VPN callee are
12 enforceable using dynamic rules. With application level
13 classification, these rules are no longer limited to fixed
14 pattern matches, such as protocol, but may be written in terms
15 of applications. An example of an application level rule
16 would be 'permit John Doe Real-Audio'. Application level
17 rules would also simplify firewall rule definitions in
18 firewall applications.

19 Step 6: After the set of code associated with the last child
20 node, which satisfied the node-criteria (referred to as the
21 packet action code) completes, a set of code associated with each
22 node in the node traversal path, is executed 623.

23 In the present embodiment, this set of code is referred to as
24 the packet post processor function(px) 625. The packet
25 postprocessing input parameters are the PBUF, the options
26 memory area and the packet action disposition. The return

1 code of the *px* function is of type *paction_t*. Examples of
2 type *paction_t* are enumerated in Figure 10.

3 Just as the packet preprocessing may decrypt a packet, the
4 packet postprocessing may perform actions such as encrypting a
5 packet 627. As the packet originally traversed the
6 classification tree, the node traversal path was created.
7 Before returning to the base operating system, in reverse node
8 traversal order, packet postprocessing is executed.

9 Normally, the packet disposition is maintained through
10 postprocessing. Only in unusual circumstances does the
11 postprocessing not follow the recommended packet action and
12 previous post processing disposition. For example, with VPN
13 tunnels the outbound tunnel may have been torn down during the
14 classification tree traversal.

15 Step 7: After the packet processing completes, control returns to
16 the base operating system, which discards, forwards, redirects or
17 locally processes the packet, based on the final disposition 633.

18 Figure 11 shows an example embodiment of the present invention
19 as an apparatus to classify and/or augment the disposition of a
20 data packet shown in Figure 11. The apparatus includes a network
21 interface device (1101) to receive a packet from the physical
22 network and pass the packet to a root node of a classification
23 tree, and the reverse, to receive a packet from the root node and
24 send a packet to the physical network. The apparatus also
25 includes a packet module (1103) to successively pass the packet
26 from child node to child node on each tree level until a first

1 child node of a tree level of the classification tree indicates a
2 satisfaction of a node-criteria of that first child node. The
3 first child node forms the data packet into a matched packet
4 until no first child node of a next level at a succeeding next
5 level indicates satisfaction of the node-criteria of the first
6 child node of the next level.

7 It is noted that an accelerator chip can be used to implement the
8 packet module (1103). This chip can be used as the basis of a
9 firewall box, a border server, or as an application level
10 classification system such as needed when diagnosing high speed
11 networking problems.

12 Other apparatus embodiments of the present invention may be
13 implemented in ways known to those familiar with the art. For
14 example, the invention may be implemented using an apparatus for
15 classifying a data packet. This apparatus includes: means for
16 receiving the data packet at a root node of a classification
17 tree; means for successively passing the data packet to each
18 child of a first tree level until a first child node of the first
19 tree level of the classification tree indicates a satisfaction of
20 a node-criteria of said first child node, and the first child
21 node forming said data packet into a matched packet; and means
22 for repeating the steps of passing and forming for a next tree
23 level until no first child node of said next tree level at a
24 succeeding next level indicates satisfaction of the node-criteria
25 of said first child node of said succeeding next level. This
26 apparatus may, for example, be in the form of a floppy or hard
27 disk, flash memory, or external magnetic media, etc.

1 Another example embodiment of the present invention is an
2 apparatus for determining disposition of a packet received at a
3 child node. This apparatus includes: an interrupt context of a
4 control program, with the child node existing within the
5 interrupt context; an external process outside of the interrupt
6 context of the control program; means for passing said packet and
7 a first disposition of said packet to the external process, the
8 external process to augment the packet disposition by employing a
9 process specific means and to return an augmented packet with an
10 augmented disposition to the child node; and the interrupt
11 context including means for receiving the augmented packet and
12 the augmented disposition from the external process. This
13 apparatus may, for example, also be in the form of a hard disk, a
14 floppy disk, or external magnetic media, etc. A control program
15 may be implemented as software that manages the example
16 apparatus.

17 The present invention can be realized in hardware, software, or a
18 combination of hardware and software. The present invention can
19 be realized in a centralized fashion in one computer system, or
20 in a distributed fashion where different elements are spread
21 across several interconnected computer systems. Any kind of
22 computer system - or other apparatus adapted for carrying out the
23 methods described herein - is suited. A typical combination of
24 hardware and software could be a general purpose computer system
25 with a computer program that, when being loaded and executed,
26 controls the computer system such that it carries out the methods
27 described herein. The present invention can also be embedded in
28 a computer program product, which comprises all the features
29 enabling the implementation of the methods described herein, and

1 which - when loaded in a computer system - is able to carry out,
2 or cause the carrying out of these methods.

3 Computer program means or computer program in the present context
4 mean any expression, in any language, code or notation, of a set
5 of instructions intended to cause a system having an information
6 processing capability to perform a particular function either
7 directly or after either or both of the following:

- 8 1. conversion to another language, code or notation; and/or
- 9 2. reproduction in a different material form.

10 It is noted that the foregoing has outlined some of the more
11 pertinent objects and embodiments of the present invention. The
12 concepts of this invention may be used for many applications.
13 Thus, although the description is made for particular
14 arrangements and methods, the intent and concept of the invention
15 is suitable and applicable to other arrangements and
16 applications. For example, although reference is made to a data
17 packet, the invention is similarly applicable to a non-data
18 packet. It will be clear to those skilled in the art that other
19 modifications to the disclosed embodiments can be effected
20 without departing from the spirit and scope of the invention.
21 The described embodiments ought to be construed to be merely
22 illustrative of some of the more prominent features and
23 applications of the invention. Other beneficial results can be
24 realized by applying the disclosed invention in a different
25 manner or modifying the invention in ways known to those familiar
26 with the art. Thus, it should be understood that the embodiments

1 has been provided as an example and not as a limitation. The
2 scope of the invention is defined by the appended claims.

1 CLAIMS

2 Having thus described our invention, what we claim as new and
3 desire to secure by Letters Patent is as follows:

4 1. A method for classifying a data packet, the method
5 comprising:

6 receiving the data packet at a root node of a classification
7 tree;

8 successively passing the data packet to each child of a first
9 tree level until a first child of the first tree level of the
10 classification tree indicates a satisfaction of a node-criteria
11 of said first child, and the first child forming said data packet
12 into a matched packet; and

13 repeating the step of passing and forming for a next tree
14 level until no first child of said next level at a succeeding
15 next level indicates satisfaction of the node-criteria of said
16 first child of said next level.

17 2. A method as recited in claim 1, wherein the step of passing
18 includes executing a set of code which returns a status
19 indication.

20 3. A method as recited in claim 1, wherein the step of forming
21 includes the first child specifying a set of code to be run
22 subsequently.

1 4. A method as recited in claim 3, wherein the step of
2 specifying includes specifying the set of code to be run
3 following satisfaction.

4 5. A method as recited in claim 1, further comprising
5 dynamically adding at least one node in at least one level of the
6 classification tree.

7 6. A method as recited in claim 5, wherein said at least one
8 new child node is a Real Audio node.

9
10 7. A method for classifying a packet, said method comprising
11 suspending a packet classification process in progress for said
12 packet; and obtaining external information employed in said
13 classifying.

14 8. A method in claim 7, wherein the step of obtaining includes
15 augmenting a node-criteria of a node in a classification tree
16 with external information.

17 9. A method as in claim 8, wherein the external information
18 includes identification of the originator of said packet.

19 10. A method as in claim 8, wherein the external information
20 includes authentication of an originator of said packet.

21 11. A method as recited in claim 7, wherein the classification
22 process is an extendible classifier process.

1 12. A method as recited in claim 1, further comprising the step
2 of parsing said matched packet and generating relevant
3 information.

4 13. A method as recited in claim 1, further comprising the step
5 of transforming said matched packet into a transformed packet.

6 14. A method as recited in claim 1, further comprising
7 associating the packet with a last first child indicating
8 satisfaction.

9 15. A method as recited in claim 14, further comprising
10 executing a set of code in accordance with said last first child.

11 16. A method as recited in claim 1, further comprising
12 determining a disposition of the data packet.

13 17. A method for determining disposition of a packet received at
14 a child node, said method comprising:
15

16 passing said packet and a first disposition of said packet to an
17 external process; and

18 said external process augmenting the packet disposition by
19 employing a process specific means; and returning the augmented
20 packet and an augmented disposition to the child node.

21 18. A method as recited in claim 17, further comprising
22 suspending a disposition process in progress for said packet.

- 1 19. A method as in claim 18, wherein the augmented disposition
2 includes identification of an originator of said packet.
- 3 20. A method as in claim 18 wherein the augmented disposition
4 includes authentication of an originator of said packet.
- 5 21. A method as recited in claim 18, wherein the disposition is
6 employed for policy enforcement.
- 7 22. A method as recited in claim 16, further comprising
8 employing the classification process as a firewall.
- 9 23. A method as recited in claim 1, further comprising employing
10 the classification process for application level classification.
- 11 24. A method as recited in claim 23, further comprising
12 employing the classification process for policy enforcement.
- 13 25. A method as recited in claim 23, further comprising
14 employing the classification process for rate limiting.
- 15 26. A method as recited in claim 23, further comprising
16 employing the classification process for load balancing.
- 17 27. A method as recited in claim 1, further comprising employing
18 the classification process to shape traffic .
- 19 28. An apparatus to classify a data packet, the apparatus
20 comprising:

1 a network interface device to receive the data packet from the
2 physical network and pass the data packet to the root node of a
3 classification tree, and the reverse, to receive the data packet
4 from the root node and send the data packet to the physical
5 network;

6 a packet module to successively pass the packet from child
7 node to child node at a next tree level until a first child node
8 of the next tree level of the classification tree which indicates
9 a satisfaction of a node-criteria of the first child node, and to
10 form the data packet into a matched packet until no first child
11 node of at a succeeding next level indicates satisfaction of the
12 first node-criteria of the first child node of the succeeding
13 next level.

14 29. An apparatus as recited in claim 28, wherein a portion of
15 the apparatus is implemented as an accelerator chip.

16 30. An apparatus as recited in claim 28, wherein the apparatus
17 is employed for application level classification.

18 31. An apparatus as recited in claim 28, wherein the apparatus
19 is employed as a firewall.

20 32. An apparatus as recited in claim 28, wherein the apparatus
21 is employed as a border server.

22 33. A method as recited in claim 2, wherein the status
23 indication is of the pm_t type.

1 34. An article of manufacture comprising a computer usable
2 medium having computer readable program code means embodied
3 therein for causing classification of a data packet, the computer
4 readable program code means in said article of manufacture
5 comprising computer readable program code means for causing a
6 computer to effect the steps of claim 1.

7 35. An article of manufacture as recited in claim 34, the
8 computer readable program code means in said article of
9 manufacture further comprising computer readable program code
10 means for causing a computer to effect dynamically adding at
11 least one node in at least one level of the classification tree.

12 36. An article of manufacture comprising a computer usable
13 medium having computer readable program code means embodied
14 therein for causing classification of a data packet, the computer
15 readable program code means in said article of manufacture
16 comprising computer readable program code means for causing a
17 computer to effect the steps of claim 8.

18 37. A computer program product comprising a computer usable
19 medium having computer readable program code means embodied
20 therein for causing a determination of a disposition of a packet,
21 the computer readable program code means in said computer program
22 product comprising computer readable program code means for
23 causing a computer to effect the steps of claim
24 18.

25 38. An apparatus for classifying a data packet, the apparatus
26 comprising:

1 means for receiving the data packet at a root node of a
2 classification tree;

3 means for successively passing the data packet to each child
4 of a first tree level until a first child node of the first tree
5 level of the classification tree indicates a satisfaction of a
6 node-criteria of said first child node, and the first child node
7 forming said data packet into a matched packet; and

8 means for repeating the steps of passing and forming for a
9 next tree level until no first child node of said next tree level
10 at a succeeding next level indicates satisfaction of the
11 node-criteria of said first child node of said succeeding next
12 level.

13 39. An apparatus for determining disposition of a packet
14 received at a child node, said apparatus comprising:

15 an interrupt context of a control program, said child node
16 existing within the interrupt context;

17 an external process outside of the interrupt context of the
18 control program;

19 means for passing said packet and a first disposition of said
20 packet to the external process, said external process to augment
21 the packet disposition by employing a process specific means and
22 to return an augmented packet with an augmented disposition to
23 the child node; and

1 said interrupt context including means for receiving said
2 augmented packet and said augmented disposition from said
3 external process.

1 NETWORK DATA PACKET CLASSIFICATION AND DEMULTIPLEXING

2 Abstract

3 The present invention provides methods and apparatus for
4 classifying and demultiplexing packets in a network protocol
5 stack. It provides extendibility for packet processing in the
6 network protocol stack by defining a standard method for adding
7 new functionality. It provides a method to obtain external
8 information, from an application scheduled outside of the
9 forwarding or interrupt context of the kernel, in order to
10 augment packet classification and/or augment packet disposition.
11 In some embodiments, external information augments a criteria of
12 a node in a classification tree with additional information. It
13 presents a way of augmenting which suspends the classification
14 process until an application, scheduled outside of the forwarding
15 or interrupt context of the kernel, completes. The resulting
16 external information is used to augment the packet
17 classification. In some embodiments of the method, the external
18 information includes authentication of an originator of the
19 packet by correlating a tunnel id with a userid, and/or using
20 s/ident for out of band authentication. The classification
21 process enables enforcement of a site policy.

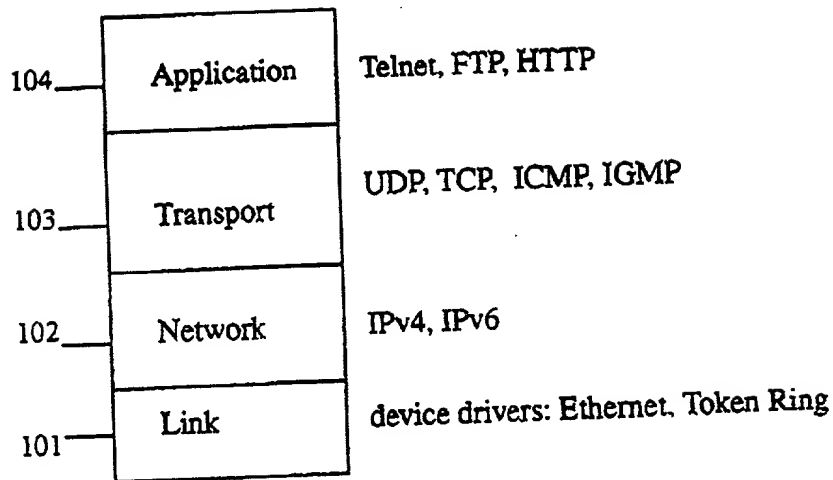


FIGURE 1.

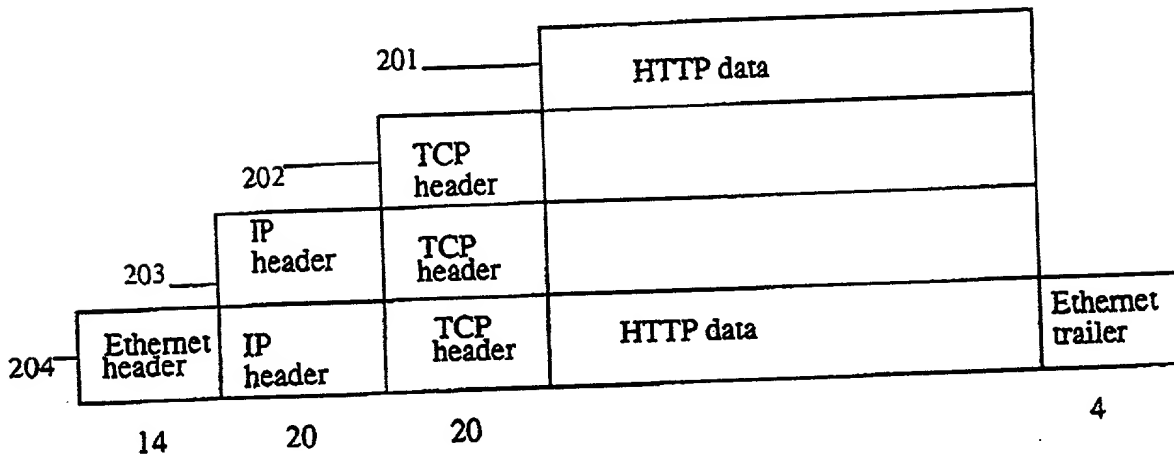


FIGURE 2.

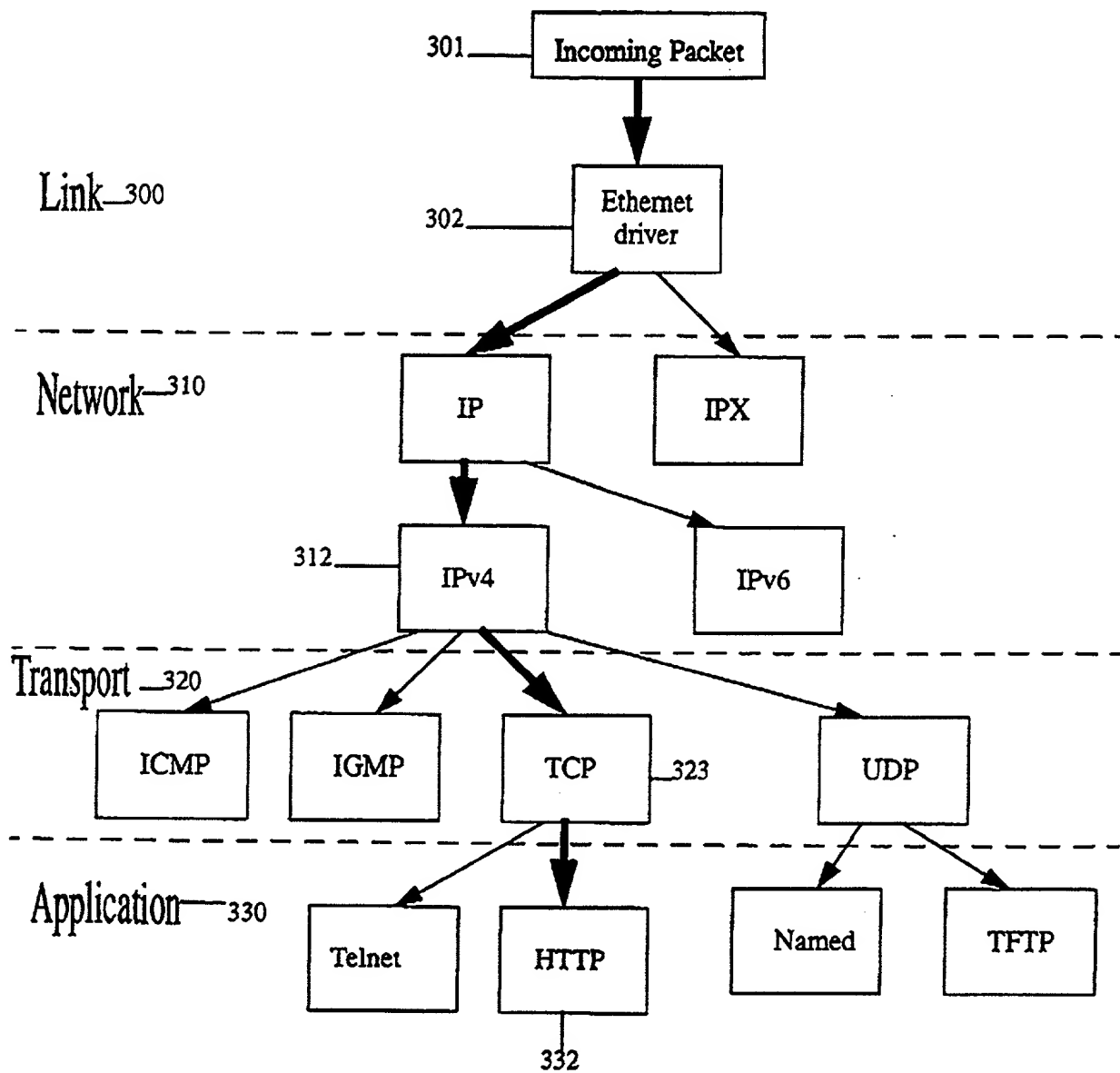


FIGURE 3.

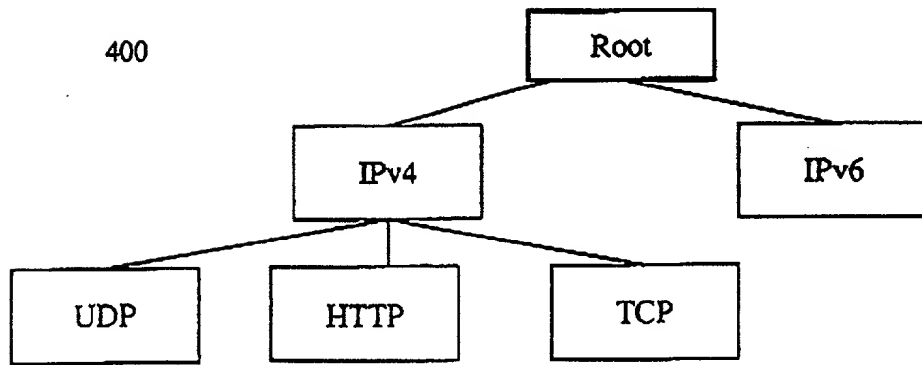


FIGURE 4.

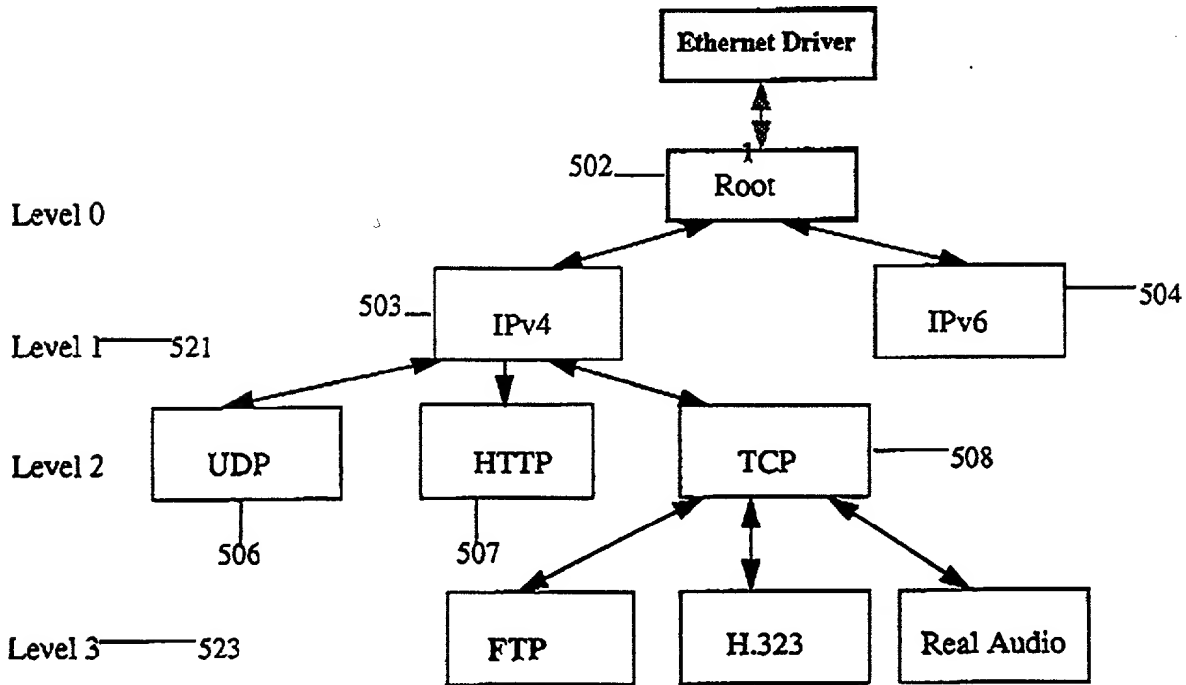


FIGURE 5.

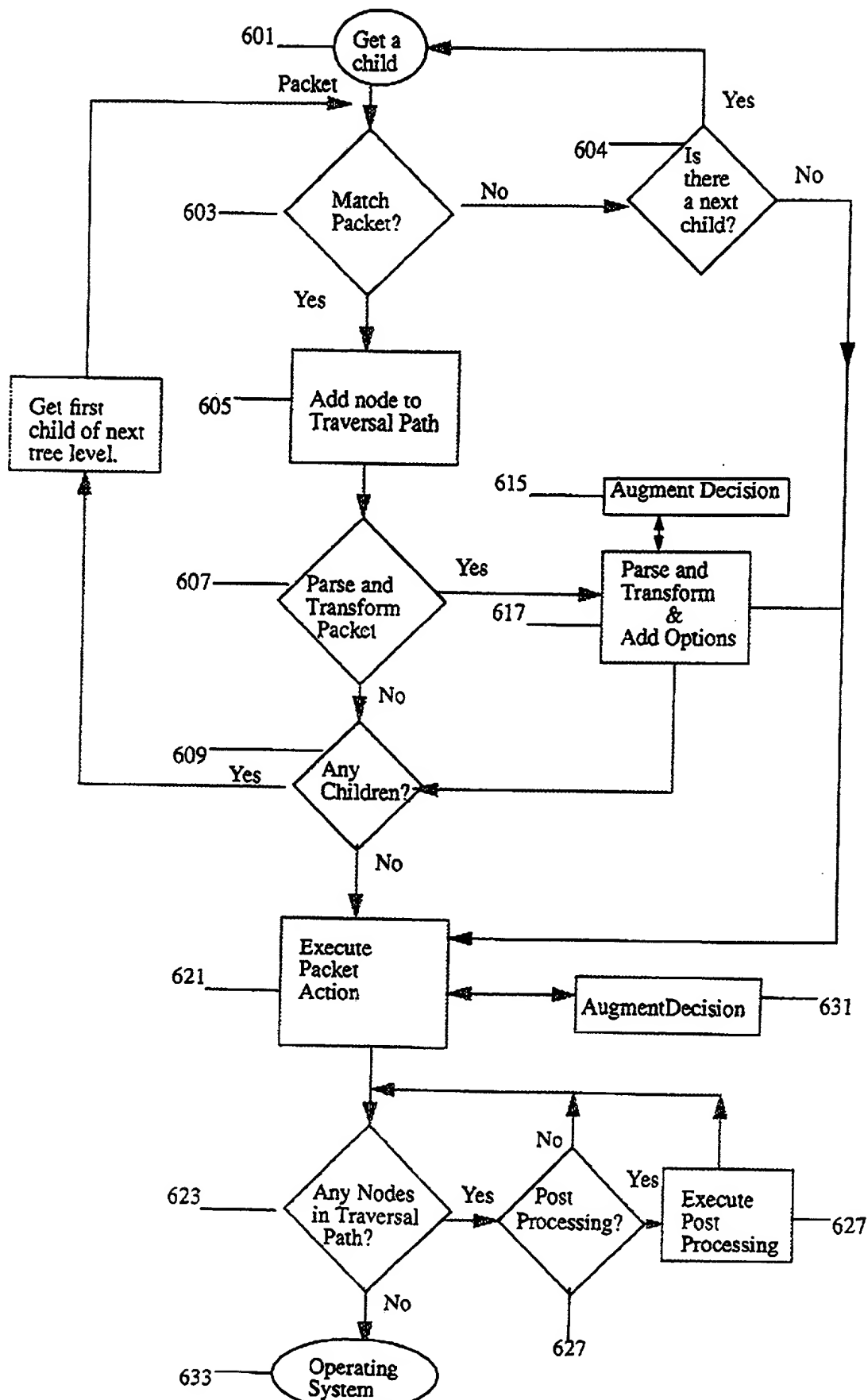


FIGURE 6.

700

No_Match - Didn't match the packet; continue normal tree traversal
 Match_OK - Matched the packet; continue normal tree traversal
 Match_This - Matched this node; execute the packet action code immediately
 Match_Discard - Matched the packet; immediately discard the packet
 Match_Forward - Matched the packet; immediately forward the packet

FIGURE 7.

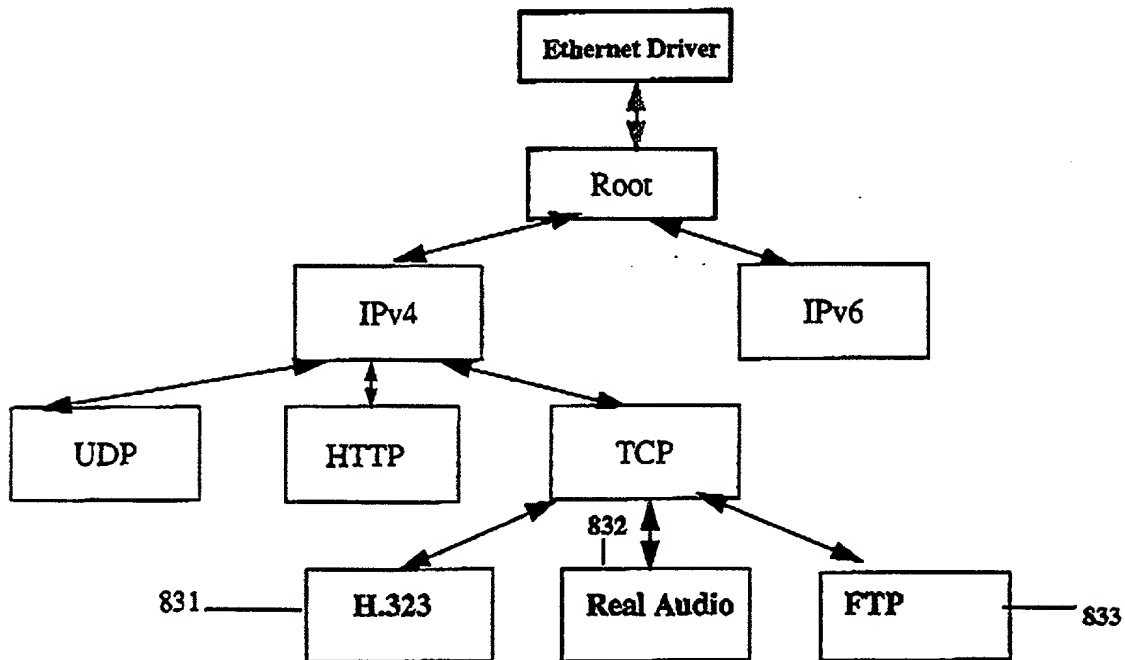


FIGURE 8.

900

PP_NorOK - Stop tree traversal
PP_OK - Continue tree traversal
PP_Discard - Immediately stop traversing the tree; discard the packet
PP_Forward - Immediately stop traversing the tree; forward the packet.

FIGURE 9.

1000

OK_Discard - Reverse traverse the tree for postprocess;
recommend to discard the packet.
OK_Forward - Reverse traverse the tree for postprocessing;
recommend to forward the packet.
Stop_Discard - Discard the packet immediately without reverse traversing the tree.
Stop_Forward - Forward the packet immediately without reverse traversing the tree.
OK_Local - Accept the packet locally.
OK_Redirect - Redirect the packet to a foreign host.

FIGURE 10.

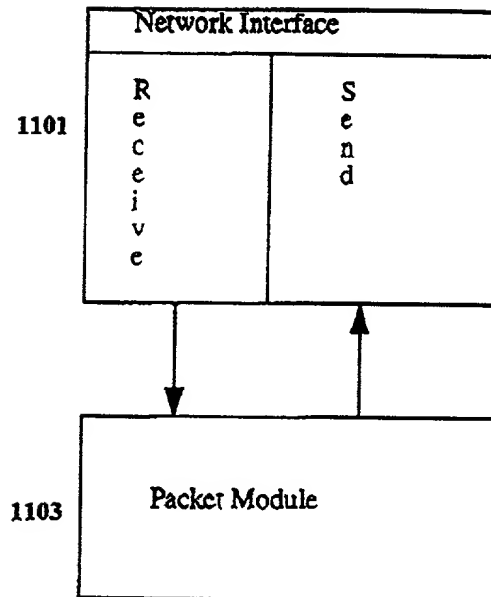


FIGURE 11.